Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 15, 2015

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
T-Systems International GmbH

January 11, 2015

Stream Schedulers and a New Data Chunk for the Stream Control Transmission Protocol
draft-ietf-tsvwg-sctp-ndata-02.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrary large user messages. However, the sender can not interleave different user messages which causes head of line blocking at the sender side. To overcome this limitation, this document adds a new data chunk to SCTP.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

When SCTP [RFC4960] was initially designed it was mainly envisioned for transport of small signaling messages. Late in the design stage it was decided to add support for fragmentation and reassembly of larger messages with the thought that someday Session Initiation Protocol (SIP) [RFC3261] style signaling messages may also need to use SCTP and a single MTU sized message would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications which might send very large messages over SCTP. When such large messages are now sent over SCTP a form of sender side head of line blocking becomes created within the protocol. This head of line blocking is caused by the use of the Transmission Sequence Number (TSN) for two different purposes:

1. As an identifier for DATA chunks to provide a reliable transfer.

2. As an identifier for the sequence of fragments to allow reassembly.

The protocol requires all fragments of a user message to have consecutive TSNs. Therefore the sender can not interleave different messages.

This document describes a new Data chunk called N-DATA. This chunk incorporates all the flags and properties of the current SCTP Data chunk but also adds a new field in its chunk header, the Fragment

Sequence Number (FSN). Then the FSN is only used for reassembly and the TSN only for the reliability. Therefore, the head of line blocking caused by the original design is avoided.

The support of the N-DATA chunk is negotiated during the association setup using the Supported Extensions Parameter as defined in [RFC5061]. If N-DATA support has been negotiated for an association N-DATA chunks are used for all user-messages and no DATA chunks. It should be noted, that an SCTP implementation needs to support the coexistence of associations using DATA chunks and associations using N-DATA chunks.

This document also defines several stream schedulers for general SCTP associations. If N-DATA support has been negotiated, several more schedulers are available.

1.2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. N-DATA Chunk

The following Figure 1 shows the new data chunk N-DATA.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Type = 64   |  Res  |I|U|B|E|           Length              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              TSN                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        Stream Identifier      |     Stream Sequence Number    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Payload Protocol Identifier                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Message Identifier                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Fragment Sequence Number                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   \                                                               \
   /                           User Data                           /
   \                                                               \
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: N-DATA chunk format

The only differences between the N-DATA chunk in Figure 1 and the DATA chunk defined in [RFC4960] and [RFC7053] is the addition of the new Message Identifier (MID) and Fragment Sequence Number (FSN).

Message Identifier (MID): 32 bits (unsigned integer)
The Message Identifier. Please note that the MID is in "network byte order", a.k.a. Big Endian.

Fragment Sequence Number (FSN): 32 bits (unsigned integer)
Identifies the fragment number of this piece of a message. FSN's are unsigned numbers, the first fragment MUST start at 0 and MUST have the 'B' bit set. The last fragment of a message MUST have the 'E' bit set. Note that the FSN may wrap completely multiple times allowing arbitrary large messages. Please note that the FSN is in "network byte order", a.k.a. Big Endian.

3. Procedures

3.1. Sender Side Considerations

A sender MUST NOT send a N-DATA chunk unless the peer has indicated its support of the N-DATA chunk type within the Supported Extensions Parameter as defined in [RFC5061]. If N-DATA support has been negotiated on an association, N-DATA chunks MUST be used for all user data and DATA-chunk MUST NOT be used at all. If N-DATA support has not been negotiated on an association, DATA chunks MUST be used for all user data and N-DATA chunks MUST NOT be used at all.

A sender MUST NOT use the N-DATA chunk unless the user has requested that use (e.g. via the socket API, see Section 4). This constraint is made since usage of this chunk requires that the application be willing to interleave messages upon reception within an association. This is not the default choice within the socket API (see [RFC6458]) thus the user MUST indicate support to the protocol of the reception of completely interleaved messages. Note that for stacks that do not implement [RFC6458] they may use other methods to indicate interleaved message support and thus enable the usage of the N-DATA chunk, the key is that the the stack MUST know the application has indicated its choice in wanting to use the extension.

Sender side usage of the N-DATA chunk is quite simple. Instead of using the TSN for fragmentation purposes, the sender uses the new FSN field to indicate which fragment number is being sent. The first fragment MUST have the 'B' bit set. The last fragment MUST have the 'E' bit set. All other fragments MUST NOT have the 'B' or 'E' bit set. If the 'I' bit is set the 'E' bit MUST also be set, i.e. the 'I' bit may only be set on the last fragment of a message. All other properties of the existing SCTP DATA chunk also apply to the N-DATA

chunk, i.e. congestion control as well as receiver window conditions MUST be observed as defined in [RFC4960].

Note that the usage of this chunk should also imply late binding of the actual TSN to any chunk being sent. This way other messages from other streams may be interleaved with the fragmented message.

The sender MUST NOT have more than one ordered fragmented message being produced in any one stream. The sender MUST NOT have more than one un-ordered fragmented message being produced in any one stream. The sender MAY have one ordered and one unordered fragmented message being produced within a single stream. At any time multiple streams MAY be producing an ordered or unordered fragmented message.

3.2. Receiver Side Considerations

Upon reception of an SCTP packet containing a N-DATA chunk if the message needs to be reassembled, then the receiver MUST use the FSN for reassembly of the message and not the TSN. Note that a non-fragmented messages is indicated by the fact that both the 'E' and 'B' bits are set. An ordered or unordered fragmented message is thus identified with any message not having both bits set.

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to allow applications to use the extension described in this document.

Please note that this section is informational only.

4.1. SCTP_ASSOC_CHANGE Notification

The notification has to be extended to provide SCTP_SUPPORTS_INTERLEAVING in sac_info if the support of N-DATA has been negotiated.

4.2. Socket Options

option name	data type	get	set
SCTP_NDATA_ENABLE	int	X	X
SCTP_PLUGGABLE_SS	struct sctp_assoc_value	X	X
SCTP_SS_VALUE	struct sctp_stream_value	X	X

4.2.1. Enable or Disable the Interleaving Capability (SCTP_NDATA_ENABLE)

A new socket option to turn on/off the usage of the N-DATA chunk. Turning this option on only effect future associations, and MUST be turned on for the protocol stack to indicate support of the N-DATA chunk to the peer during association setup. Turning this option off, will prevent the N-DATA chunk from being indicated supported in future associations, and will also prevent current associations from producing N-DATA chunks for future large fragmented messages. Note that this does not stop the peer from sending N-DATA chunks.

An N-DATA chunk aware application should also set the fragment interleave level to 2. This allows the reception from multiple streams simultaneously. Failure to set this option can possibly lead to application deadlock.

4.2.2. Get or Set the Stream Scheduler (SCTP_PLUGGABLE_SS)

A stream scheduler can be selected with the SCTP_PLUGGABLE_SS option for setsockopt(). The struct sctp_assoc_value is used to specify the association for which the scheduler should be changed and the value of the desired algorithm.

The definition of struct sctp_assoc_value is the same as in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

assoc_value: This specifies which scheduler is used. The following constants can be used:

SCTP_SS_DEFAULT: The default scheduler used by the SCTP implementation. Typical values are SCTP_SS_ROUND_ROBIN or SCTP_SS_FIRST_COME.

SCTP_SS_ROUND_ROBIN: This scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues.

SCTP_SS_ROUND_ROBIN_PACKET: This is a round-robin scheduler but only bundles user messages of the same stream in one packet. This minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

SCTP_SS_PRIORITY: Scheduling with different priorities is used. Streams having a higher priority will be scheduled first and when multiple streams have the same priority, the default scheduling should be used for them. The priority can be assigned with the sctp_stream_value struct. The higher the assigned value, the lower the priority, that is the default value 0 is the highest priority and therefore the default scheduling will be used if no priorities have been assigned.

SCTP_SS_FAIR_BANDWITH: A fair bandwidth distribution between the streams can be activated using this value. This scheduler considers the lengths of the messages of each stream and schedules them in a certain way to maintain an equal bandwidth for all streams.

SCTP_SS_WEIGHTED_ROUND_ROBIN: A weighted round robin distribution between the streams can be activated using this value. This scheduler considers the lengths of the messages of each stream and schedules them in a certain way to use the bandwidth according to the given weights.

SCTP_SS_FIRST_COME: The simple first-come, first-serve algorithm is selected by using this value. It just passes through the messages in the order in which they have been delivered by the application. No modification of the order is done at all.

4.2.3. Get or Set the Stream Scheduler Parameter (SCTP_SS_VALUE)

Some schedulers require additional information to be set for single streams as shown in the following table:

name	per stream info
SCTP_SS_DEFAULT	no
SCTP_SS_RR	no
SCTP_SS_RR_INTER	no
SCTP_SS_RR_PKT	no
SCTP_SS_RR_PKT_INTER	no
SCTP_SS_PRIO	yes
SCTP_SS_PRIO_INTER	yes
SCTP_SS_FB	no
SCTP_SS_FB_INTER	no
SCTP_SS_WRR	yes
SCTP_SS_WRR_INTER	yes
SCTP_SS_FCFS	no

This is achieved with the SCTP_SS_VALUE option and the corresponding struct sctp_stream_value. The definition of struct sctp_stream_value is as follows:

```
struct sctp_stream_value {
  sctp_assoc_t assoc_id;
  uint16_t stream_id;
  uint16_t stream_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

stream_id: Holds the stream id for the stream for which additional information has to be provided.

stream_value: The meaning of this field depends on the scheduler specified. It is ignored when the scheduler does not need additional information.

5. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

[NOTE to RFC-Editor:

The suggested values for the chunk type and the chunk flags are tentative and to be confirmed by IANA.

]

This document (RFCXXXX) is the reference for all registrations described in this section.

A new chunk type has to be assigned by IANA. IANA should assign this value from the pool of chunks with the upper two bits set to '01'. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
64	New DATA chunk (N-DATA)	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially given by the following table:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFCXXXX]
0x02	B bit	[RFCXXXX]
0x04	U bit	[RFCXXXX]
0x08	I bit	[RFCXXXX]
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960] and [RFC6458].

7. Acknowledgments

The authors wish to thank Karen E. Egede Nielsen, Irene Ruengeler, and Lixia Zhang for her invaluable comments.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, January 2011.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, November 2013.

8.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
US

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
DE

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
T-Systems International GmbH
Fasanenweg 5
70771 Leinfelden-Echterdingen
DE

Email: rfc@robin-seggelmann.com